Lab Exercise
CPE 357
Systems Programming

CAL POLY
SAN LUIS OBISPO

LAB 3

Exercise:

Lets speed up working on a file:
Read any bitmap file and change its color grading. However, do this in 2 processes to speed up the CPU worktime.
Measure the time with and without fork() and print them, hence, do the same process twice in a row: One time with fork, one time without it.

Howto:

Similar to program 1, read a bitmap (BMP) file into the memory. Allocate your memory with mmap() and don't forget to set the shared memory flag.
Fork the process with fork().
Check which process you are. Are you the parent process? Work on the upper half of all pixels. Child? Work on the lower half. Having an odd number of rows? Catch that problem!
Measure the time with and without fork() and print the result time!

Program call:
./yourprogram [IMAGEFILE] [COLOR GRADING] [OUTPUTFILE]
[IMAGEFILE] that's your bitmap
[COLOR GRADING] three float number between 0 and 1 representing red, green and blue (RGB)
[OUTPUTFILE] the output file

Example:
./colorgrading lion.bmp 0.8 1.0 0.8 result.bmp

How we test / requirements

Very simple: We run your code with and without fork(), check the time gained and check the output image. We also check if you used a function(s) for the "work on data array" part.

Submission:

Submit the source code file, the executable and the bitmap zipped to: MYNAME_colorgrading**.c or .cpp**

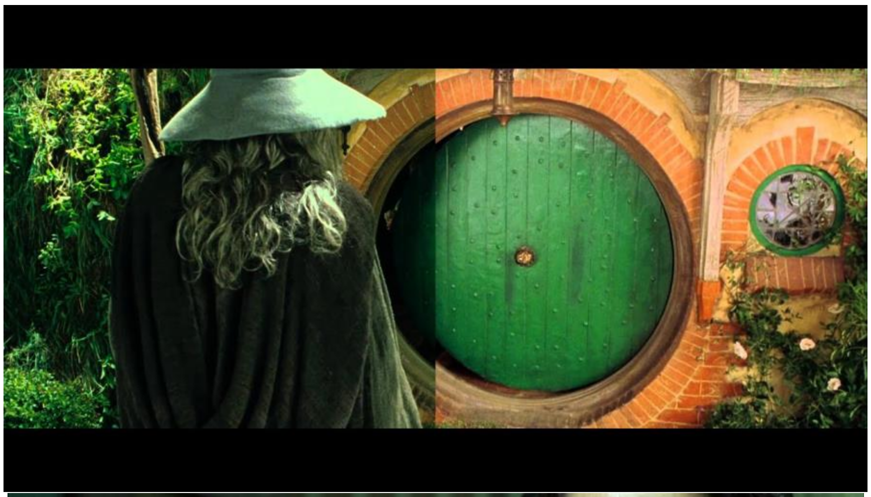
For the color grading:

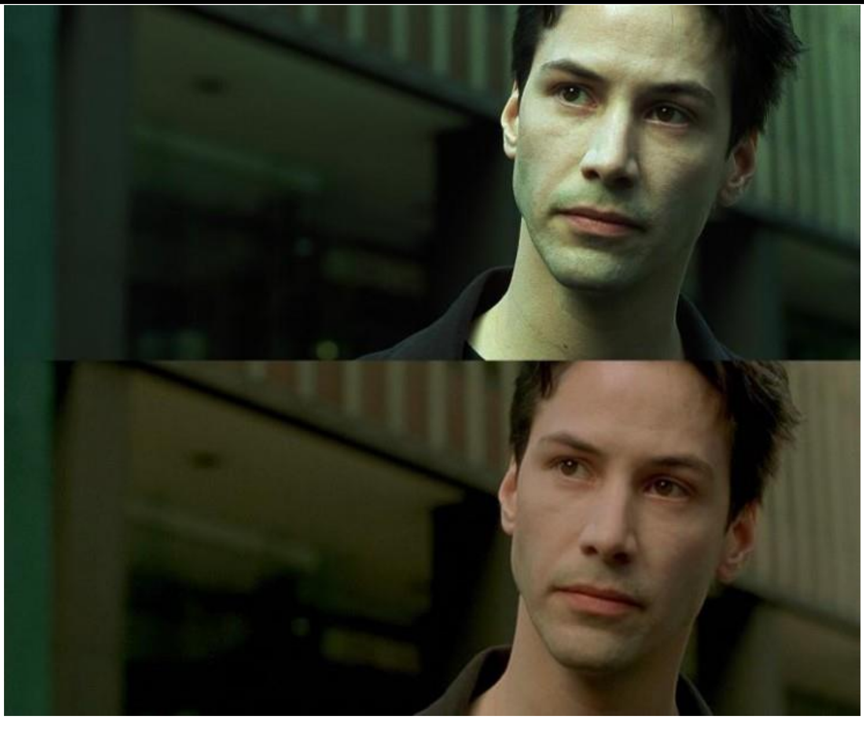
Fear not, it's not that hard.

1. You normalize (similar to lab 2) the colors, this time make sure to catch blue, green and red separately, so you get for each pixel a floating point nromalized value like
float noramlized_blue, noramlized_green, noramlized_red;
2. Multiply noramlized_blue with the blue value from the command line, do this also with green and red.
3. Get it back into [0.255] and store the value in the data array.

For those firm with math, it's just the dot product, and the dot product "checks" the similarity of values. For example, if my color grading is 1.0 0.5 0.5 (RGB), then the result image will be very (!) red-ish.

Background: Color grading is MASSIVELY important for film and photos. One (infamous) example is Lord of the Rings on DVD, or The Matrix trilogy, which oversaturated green colors, see image. I think (!) there its about 0.9 1.0 0.93. Color grading is used in every movie. It can enchance the overall feeling which the director intended to transport in a scene.





HINTS:

As always with programs taking command line arguments, do NOT start with taking command line args. Hardcode them. You will need your debugger!
This program requires multiple CPU cores.
It's pretty certain your laptop/desktop has this, but you need to make sure to activate these resources in your virtual solution. A server, like the Cal Poly server, WILL NOT let you take more than one core.